1.x — Chapter 1 summary and quiz

ALEX[1] MARCH 12, 2024

Chapter Review

A **statement** is a type of instruction that causes the program to perform some action. Statements are often terminated by a semicolon.

A **function** is a collection of statements that execute sequentially. Every C++ program must include a special function named *main*. When you run your program, execution starts at the top of the *main* function.

In programming, the name of a function (or object, type, template, etc...) is called its **identifier**.

The rules that govern how elements of the C++ language are constructed is called **syntax**. A **syntax error** occurs when you violate the grammatical rules of the language.

Comments allow the programmer to leave notes in the code. C++ supports two types of comments. Line comments start with a `//` and run to the end of the line. Block comments start with a `/*` and go to the paired `*/` symbol. Don't nest block comments.

You can use comments to temporarily disable lines or sections of code. This is called commenting out your code.

Data is any information that can be moved, processed, or stored by a computer. A single piece of data is called a **value**. Common examples of values include letters (e.g. `a`), numbers (e.g. `5`), and text (e.g. `Hello`).

A variable is a named piece of memory that we can use to store values. In order to create a variable, we use a statement called a **definition statement**. When the program is run, each defined variable is **instantiated**, which means it is assigned a memory address.

A **data type** tells the compiler how to interpret a piece of data into a meaningful value. An **integer** is a number that can be written without a fractional component, such as 4, 27, 0, -2, or -12.

Copy assignment (via operator=) can be used to assign an already created variable a value.

The process of specifying an initial value for an object is called **initialization**, and the syntax used to initialize an object is called an **initializer**.

Simplified, C++ supports 6 basic types of initialization:

Initialization Type	Example	Note
Default initialization	int x;	In most cases, leaves variable with indeterminate value
Copy initialization	int x = 5;	
Direct initialization	int x (5);	
Direct list initialization	int x { 5 };	Narrowing conversions disallowed
Copy list initialization	int x = { 5 };	Narrowing conversions disallowed
Value initialization	int x {};	Usually performs zero initialiation

Direct initialization is sometimes called parenthesis initialization, and list initialization (including value initialization) is sometimes called uniform initialization or brace initialization. You should prefer brace initialization over the other initialization forms, and prefer initialization over assignment.

Although you can define multiple variables in a single statement, it's better to define and initialize each variable on its own line, in a separate statement.

std::cout and `operator<<` allow us to output the result of an expression to the console.

std::endl outputs a newline character, forcing the console cursor to move to the next line, and flushes any pending output to the console. The `'\n'` character also outputs a newline character, but lets the system decide when to flush the output. Be careful not to use `'/n'` (forward slash).

std::cin and `operator>>` allow us to get a value from the keyboard.

A variable that has not been given a value is called an **uninitialized variable**. Trying to get the value of an uninitialized variable will result in **undefined behavior**, which can manifest in any number of ways.

C++ reserves a set of names called **keywords**. These have special meaning within the language and may not be used as variable names.

A **literal constant** is a fixed value inserted directly into the source code. Examples are 5 and "Hello world!".

An **operation** is a process involving zero or more input values, called **operands**. The specific operation to be performed is denoted by the provided **operator**. The result of an operation produces an output value.

Unary operators take one operand. **Binary** operators take two operands, often called left and right. **Ternary** operators take three operands. **Nullary** operators take zero operands.

An **expression** is a sequence of literals, variables, operators, and function calls that are evaluated to produce a single output value. The calculation of this output value is called

evaluation. The value produced is the **result** of the expression.

An **expression statement** is an expression that has been turned into a statement by placing a semicolon at the end of the expression.

When writing programs, add a few lines or a function, compile, resolve any errors, and make sure it works. Don't wait until you've written an entire program before compiling it for the first time!

Focus on getting your code working. Once you are sure you are going to keep some bit of code, then you can spend time removing (or commenting out) temporary/debugging code, adding comments, handling error cases, formatting your code, ensuring best practices are followed, removing redundant logic, etc…

First-draft programs are often messy and imperfect. Most code requires cleanup and refinement to get to great!

Quiz time

Question #1

What is the difference between initialization and assignment?

Hide Solution (javascript:void(0))[2]

> Initialization provides a variable with an initial value (at the point of creation). Assignment gives a variable a new value after the variable has already been defined.

Question #2

When does undefined behavior occur? What are the consequences of undefined behavior?

Hide Solution (javascript:void(0))[2]

> Undefined behavior occurs when the programmer does something that is ill-specified by the C++ language. The consequences could be almost anything, from crashing to producing the wrong answer to working correctly anyway.

Question #3

Write a program that asks the user to enter a number, and then enter a second number. The program should tell the user what the result of adding and subtracting the two numbers is.

The output of the program should match the following (assuming inputs of 6 and 4):

```
Enter an integer: 6
Enter another integer: 4
6 + 4 is 10.
6 - 4 is 2.
```

Hint: To print a period and a newline, use `".\n"`, not `'.\n'`.

Hide Solution (javascript:void(0))[2]

```
#include <iostream>

int main()
{
    std::cout << "Enter an integer: ";
    int x{};
    std::cin >> x;

    std::cout << "Enter another integer: ";
    int y{};
    std::cin >> y;

    std::cout << x << " + " << y << " is " << x + y << ".\n";
    std::cout << x << " - " << y << " is " << x - y << ".\n";

    return 0;
}
```

Next lesson

2.1 Introduction to functions

3

Back to table of contents

4

Previous lesson

1.11 Developing your first program

5

6

B U URL INLINE CODE C++ CODE BLOCK HELP!

Leave a comment...

Name*

Email*

Find a mistake? Leave a comment above!

Avatars from https://gravatar.com/[9] are connected to your provided email address.

Notify me about replies:

POST COMMENT

1K COMMENTS Newest

New David

April 14, 2024 7:32 pm

```
1  int main()
2  {
3      cout << "Enter an integer:";
4
5      int num{ }, num2{};
6      cin >> num;
7
8      cout << "Enter another number:";
9      cin >> num2;
10
11     cout << num << " + " << num2 << " is " << num + num2 << ".\n";
12     cout << num << " - " << num2 << " is " << num - num2 << ".\n";
13 }
```

Last edited 15 hours ago by New David

0 Reply

Kenneth.O

April 13, 2024 12:46 pm

```
#include <iostream>

using namespace std;

int main()
{

  cout << "Enter an integer: ";
  int num1{};
  cin >> num1;

  cout << "Enter another integer: ";
  int num2{};
  cin >> num2;

  cout << num1 << " + " << num2 << " is " << num1 + num2 << ".\n";
  cout << num1 << " - " << num2 << " is " << num1 - num2 << ".\n";



    return 0;
}
```

0 Reply

:DDDD

April 12, 2024 12:51 pm

I know a little bit of python but c++ is another world! Thanks for this tutorial!

```
#include <iostream>

int main()
{
    int a{};
    int b{};

    std::cout << "Enter an integer: ";
    std::cin >> a;

    std::cout << "Enter another integer: ";
    std::cin >> b;

    std::cout << a << " + " << b << " is " << a + b << ".\n";
    std::cout << a << " - " << b << " is " << a - b << ".\n";


}
```

1 Reply

Bruhhh

April 11, 2024 10:31 am

#include<iostream>
using namespace std;
int main(){
int a,b;
cin >> a >> b;
cout << "Enter an integer: " << a << ".\n";
cout << "Enter another integer: " << b << ".\n";
cout << a << " + " << b <<""<< " is " << a+b << ".\n";
cout << a << " - " << b <<""<< " is " << a-b << ".\n";
return 0;
}

0 Reply

Dakota
April 9, 2024 9:18 pm

```cpp
#include <iostream>

int main()
{
    int num{ }, num2{ };        //define variables num and num2 as integer variables

    std::cout << "Enter an integer: ";   //asking user to input an initial integer
    std::cin >> num;                     //get integer value for num from user input

    std::cout << "Enter another integer: ";   //asking user to enter another integer
    std::cin >> num2;                         //get integer value for num2 from user input

    std::cout << num << " + " << num2 << " is " << num + num2 << ".\n";
//will provide user with results using operator+ between the two integers
    std::cout << num << " - " << num2 << " is " << num - num2 << ".\n";
//will provide user with results using operator- between the two integers

    return 0;
}
```

1 Reply

Linus
April 5, 2024 6:12 pm

#include <iostream>

int main()
{
int a,b;
std::cout << "Enter an integer:\n";
std::cin >> ("%d", a);
std::cout << "Enter another integer:\n";
std::cin >> ("%d", b);
std::cout << a << " + " << b << " is " << a + b << "\n";
std::cout << a << " - " << b << " is " << a - b << "\n";
return 0;
}

0 Reply

Kojo Bailey
April 4, 2024 12:23 am

No `std::print` yet :(

```
#include <iostream>
#include <format>

int main() {
    int a{}, b{}; // Two integers for user input.
    std::cout << "Enter an integer: ";
    std::cin >> a;
    std::cout << "\n";
    std::cout << "Enter another integer: ";
    std::cin >> b;
    std::cout << "\n";

    std::cout << std::format("{} + {} is {}.\n", a, b, a + b);
    std::cout << std::format("{} - {} is {}.\n", a, b, a - b);
}
```

0 Reply

Mohamed
April 3, 2024 5:08 am

//Write a program that asks the user to enter a number,
// and then enter a second number.
// The program should tell the user what the result of adding and subtracting the two numbers is.
#include <iostream>
using namespace std;
int main()

{
int x{};
int y{};
cout<<"Entre an integer"<<'\n';
cin>>x;
cout<<"Entre another integer"<<'\n';
cin>>y;
int addition {x+y};
int subtraction {x-y};
cout<<x<<" "<<"+"<<" "<<y<<" "<<"is"<<" "<<addition<<".\n";
cout<<x<<" "<<"-"<<" "<<y<<" "<<"is"<<" "<<subtraction<<".";
}

0 Reply

Fawaz
April 1, 2024 7:58 am

```cpp
#include <iostream>

int main()
{

    int num1{};
    int num2{};

    std::cout << "Enter an integer: ";
    std::cin >> num1;

    std::cout << "Enter another integer: ";
    std::cin >> num2;

    std::cout << num1 << " + " << num2 << " is " << (num1 + num2) << ".\n";
    std::cout << num1 << " - " << num2 << " is " << (num1 - num2) << ".\n";

    return 0;

}
```

2 Reply

faraz
March 25, 2024 8:06 pm

#include <iostream>

int main()
{

```
int num1{};
int num2{};

std::cout << "Enter an integer: ";
std::cin >> num1;

std::cout << "Enter another integer: ";
std::cin >> num2;

std::cout << num1 << " + " << num2 << " is " << num1 + num2 << "\n";

std::cout << num1 << " - " << num2 << " is " << num1 - num2;

return 0;
}
```

0 Reply

jeffrey
March 24, 2024 9:26 pm

```
int expressionProgrammThatDoesAdditionAndSubstraction()
{
int num1 {};
int num2 {};
std::cout << "Enter Integer One: ";
std::cin >> num1;
std::cout << "Enter Integer Two: ";
std::cin >> num2;
int numAddition {num1 + num2};
int numSubstraction {num1 - num2};
std::cout << "Addition: " << num1 << " + " << num2 << " = " << numAddition << '\n';
std::cout << "Substraction: " << num1 << " - " << num2 << " = " << numSubstraction << '\n';
return 0;
}
int main()
{
expressionProgrammThatDoesAdditionAndSubstraction();
}
```

0 Reply

regrif07
March 11, 2024 10:52 am

I think the newline character (`'\n'`) and the fact that `std::endl` flushes the buffer are worth mentioning in the section about io operations.

1 Reply

Alex Author

Reply to regrif07 [10] March 12, 2024 5:56 pm

Added. Thanks for the suggestion.

2 Reply

Caleb

March 11, 2024 10:28 am

this is mine

For this one i made it so that they add the number then after they press enter they can subsequently add another one

```
#include <iostream>

int main()
{
int numberone{};
int numbertwo{};

std::cout << "Enter two numbers, I will add them for you \n";

std::cin >> numberone; '\n';
std::cin >> numbertwo; '\n';

int sumofproducts{ numberone + numbertwo };
std::cout << sumofproducts;
}
```

0 Reply

Ricky

Reply to Caleb [11] March 13, 2024 3:51 pm

The variable name for 'sumOfProducts' can be a bit confusing as products usually means multiplication is involved. So for a non programmer it can give off the idea that multiplication is involved.

1 Reply

JPerk
March 8, 2024 11:26 pm

```
#include <iostream>

// Program that is going to accept two values from the user and then and and
// subtract them. It will then provide both answers.

int main() {

    std::cout << " Enter first digit: ";
    int firstnum{};
    std::cin >> firstnum;

    std::cout << " Enter second digit: ";
    int secondnum{};
    std::cin >> secondnum;

    std::cout << firstnum << " + " << secondnum << " = " << firstnum +
secondnum << ".\n";
    std::cout << firstnum << " - " << secondnum << " = " << firstnum -
secondnum << ".\n";

    return 0;
}
```

0 Reply

beginner123
March 4, 2024 11:21 am

#include <iostream>

using namespace std;

int num{};
int num2{};

int main()
{
cout << "Enter an Integer: ";
cin >> num;
cout << "Enter another Integer: ";
cin >> num2;
cout << num << " + " << num2 << " is " << num + num2 << "." << endl;
cout << num << " - " << num2 << " is " << num - num2 << "." << endl;

return 0;
}

0 Reply

User
March 4, 2024 1:17 am

```
#include <iostream>

int main()
{
    int num{};
    int num2{};

    std::cout << "Enter an integer: ";
    std::cin >> num;

    std::cout << "Enter another integer: ";
    std::cin >> num2;

    std::cout << num << " + " << num2 << " is " << num + num2 << ".\n";
    std::cout << num << " - " << num2 << " is " << num - num2 << ".\n";

    return 0;
}
```

Last edited 1 month ago by User

0 Reply

Sriram R
March 1, 2024 3:32 am

#include <iostream>

int main()
{
int a;
int b;
cout<<"Enter an integer ";
cin>>a;
cout<<"Enter another integer ";
cin>>b;
cout <<a<<" + "<<b<< " is "<<a+b;
cout <<a<<" - "<<b<< " is "<<a-b;

}

0 Reply

Anonymous
February 27, 2024 8:46 pm

```cpp
#include <iostream>

int main() {
    int integer1;
    int integer2;

    std::cout << "Enter an integer: ";
    std::cin >> integer1;
    std::cout << "Enter another integer: ";
    std::cin >> integer2;
    int additionResult = integer1 + integer2;
    int subtractionResult = integer1 - integer2;
    std::cout << integer1 << " + " << integer2 << " is " << additionResult
<< ".\n";
    std::cout << integer1 << " - " << integer2 << " is " <<
subtractionResult << ".\n";

    return 0;
}
```

Last edited 1 month ago by Anonymous

1 Reply

Kelzeck

February 27, 2024 9:47 am

#include <iostream>

int main()

{

int x{};

int y{};

std::cout << "Enter an integer: ";

std::cin >> x;

std::cout << "Enter another integer: ";

std::cin >> y;

std::cout << x << " + " << y << " is " << x + y << ".\n";

std::cout << x << " - " << y << " is " << x - y << ".\n";

return 0;

}

0 Reply

jiiiong

February 27, 2024 4:36 am

A Possible Mistake.

Before everything else, thank you for this invaluable tutorials.

But I have a problem with the definition of "initializer".
You say "The syntax used to initialize a variable is called an initializer", that is an initializer is the syntax.

There is a concept called "initializer list" in latter course. A list of syntax sounds wired, while a list of values, produced by expression ， are making more sense.

So, I wonder if "**The expression in the initialization syntax that produce the initial value**" would be a better definition for "initializer".

Last edited 1 month ago by jiiiong

0 Reply

Alex Author

Reply to jiiiong [12] February 28, 2024 7:55 pm

An initializer and an initializer list are formally defined concepts -- the fact that the latter has the word "list" in the name does not imply it's just a list of the former.

Although your proposed definition works in most cases, it doesn't work in all of them. Consider value initialization -- `{}` is the initializer, but there is no expression in this syntax.

0 Reply

Links

1. https://www.learncpp.com/author/Alex/
2. javascript:void(0)
3. https://www.learncpp.com/cpp-tutorial/introduction-to-functions/
4. https://www.learncpp.com/
5. https://www.learncpp.com/cpp-tutorial/developing-your-first-program/
6. https://www.learncpp.com/chapter-1-summary-and-quiz/
7. https://www.learncpp.com/cpp-tutorial/introduction-to-the-preprocessor/
8. https://www.learncpp.com/cpp-tutorial/introduction-to-fundamental-data-types/
9. https://gravatar.com/
10. https://www.learncpp.com/cpp-tutorial/chapter-1-summary-and-quiz/#comment-594539
11. https://www.learncpp.com/cpp-tutorial/chapter-1-summary-and-quiz/#comment-594535
12. https://www.learncpp.com/cpp-tutorial/chapter-1-summary-and-quiz/#comment-594079